(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 489 594 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **91311348.6**

(51) Int. Cl.^{5} : **G06F 15/72**

(22) Date of filing : **05.12.91**

(30) Priority : **06.12.90 US 624163**

(43) Date of publication of application :
10.06.92 Bulletin 92/24

(84) Designated Contracting States :
DE FR GB IT

(71) Applicant : **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504 (US)

(72) Inventor : **Albaugh, Virgil Anthony**
27 Woodland Loop
Round Rock, Texas 78664 (US)
Inventor : **Urquhart, Robert John**
9210 Mystic Oaks Trail
Austin, Texas 78750 (US)

(74) Representative : **Mitchell, Allan Edmund et al**
IBM United Kingdom Limited Intellectual Property Department Hursley Park
Winchester Hampshire SO21 2JN (GB)

(54) Computer graphics system.

(57) In a computer graphics system, a means and method is provided for initializing and updating a group of pixels contained on a display in blocks. A group of pixels is considered as a block and has a status word associated therewith. This status word maintains a running total of the maximum Z value of any pixel contained within a group or block of pixels. In this manner, once a block of pixels is rendered on to the display screen, a comparison can be made between the current pixels being displayed and a group of pixels which are to be displayed. The minimum Z value of the block of pixels to be displayed is compared with the maximum Z value for the block of pixels currently being displayed. If the current maximum Z value, as stored in the status word, is less than the minimum Z value for the pixels to be displayed, then the block of pixels currently being displayed will all "win" when compared to the pixels in the block to be displayed. In this case, a full block bypass of the blocks of pixels to be displayed is implemented, thereby saving considerable time and overhead when compared to conventional Z buffer systems that compare Z values for each individual pixel.

EP 0 489 594 A2

The present invention relates to computer graphics systems having a display with a plurality of pixels thereon and a frame buffer, and to methods of displaying an object thereon.

The frame buffer includes Z buffer which is a depth buffer holding a value associated with the depth of each pixel on a display screen and is generally used to remove hidden surfaces or lines when rendering 3-D solids on a display. Conventional Z buffers include one word associated with each displayed pixel. The present invention includes a method of improving both the initialization and updating functions associated with the Z buffer. More particularly, pixels are grouped together in blocks and a status word (Za) is associated with each block. This status word contains the maximum value of any pixel within the block. It should be noted that maximum value refers to maximum depth of a pixel within a display screen, i.e. the maximum distance from the eye of a viewer to the pixel.

Conventional Z buffers initialize each pixel position individually when clearing the Z buffer, and updating the Z buffer for new values. That is, at initialization each pixel is set to a Z max value such that any new Z value will be closer to a view of the display than the Z max value. Therefore, any subsequent Z value will "win" when compared to Z max.

To update the Z buffer of conventional systems, the existing value in the Z buffer for each pixel is compared with the new Z value to be written to the same pixel. If the new Z value is less than the value currently in the Z buffer for that pixel, then the new Z value replaces the existing value in the Z buffer. In this manner, pixel representing objects closer to a view of the display are maintained in the Z buffer and objects further from the viewer are obscured. If the existing value in the Z buffer is less than the new value for the object being displayed then the existing value remains in the Z buffer. This procedure is applied to each pixel of each scan-converted object until the entire scene is rendered.

In contrast to the prior art, the present invention provides for setting the Za word to a maximum Z value (Z max) such that subsequent Z values for each pixel in the block associated with Az will win (be displayed) when compared with each pixel in the block under consideration. In other words, instead of setting each pixel to Z max, only the Za word is set to Z max and all the pixels in that block are considered to be initialized to Z max. In this manner significant time and overhead is saved when initializing a Z buffer. These Za words are initially set to a negative value which indicates that the block has been "logically cleared", which means that actual Z values have yet to be determined and written to the Z buffer. The present invention also provides a method which eliminates the need to apply a comparison between a new Z value and an existing Z value for each individual pixel on a display screen.

According to the invention, therefore, there is provided a method of displaying at least one object on a computer graphics system having a display with a plurality of pixels thereon and a frame buffer, the method comprising the steps of associating a portion of the plurality of pixels together in a block, storing a maximum depth value for each blocs, determining whether the object to be displayed intersects the block, and drawing all the pixels within the block to the display.

The invention extends to a method of displaying at least one object on a computer graphics system having a display with a plurality of pixels thereon and a frame buffer, the method comprising the steps of associating a portion of the plurality of pixels together in a block, storing a maximum depth value for each block, determining whether the object to be displayed intersects the block, determining an estimated minimum Z value for a block of pixels corresponding to the object to the displayed, comparing the stored maximum depth value for the block of pixels corresponding to the object currently being displayed with the estimated minimum Z value, and continuing to display the current block of pixels when the stored maximum depth value is less than the estimated minimum Z value.

Advantageously, the step of continuing to display comprises a method wherein the step of continuing to display comprises the step of bypassing consideration of the block of pixels corresponding to the object to be displayed.

The invention further provides a computer graphics system having a display with a plurality of pixels thereon and a frame buffer, comprising means for associating a portion of the plurality of pixels together in a block, means for storing a maximum depth value for each block, means for determining whether an object to be displayed interests the block, and means for drawing the pixels within the block to the display.

From another aspect, the invention provides a computer graphics system having a display with a plurality of pixels thereon and a frame buffer, comprising means for associating a portion of the plurality of pixels together in a block, means for storing a maximum depth value for each block, means for determining whether an object to be displayed intersects the block, means for determining an estimated minimum Z value for a block of pixels corresponding to the object to be displayed, means for comparing the stored maximum depth value for the block of pixels corresponding to an object currently being displayed with the estimated minimum Z value, and means for maintaining the display of the current block of pixels when the stored maximum Z value is less than the estimated minimum Z value.

Other features of the invention are defined in the

appended claims.

How the invention can be carried into effect is hereinafter particularly described with reference to the accompanying drawings in which :-

Figure 1 shows a display as part of a system according to the invention, having a block of associated pixels;

Figure 2 shows a portion of the system and the display with three blocks of pixels and a line to be displayed;

Figures 3A and 3B form a flow chart representing the operation of a system according to the present invention;

Figure 4 is a graphical representation of the slope of lines to be drawn and their comparison with the status word Za;

Figure 5 is a listing of pseudo-code which is an example of one means of implementing the present invention; and

Figure 6 is a schematic diagram showing a computer graphics system on which the present invention may be implemented.

A computer graphics system (Figure 6) which may utilize the present invention includes a display or monitor 10, such as a CRT or the like, with digital-to-analog converter (DAC) 20 providing a signal to be displayed thereon. A graphics adapter card 30, or the like, includes frame buffer 22, Z buffer 24 and its associated status word (Za) 26, and rendering hardware 33. A central processing unit (CPU) 31 is provided along with operating system 32 which will contain the actual lines of programming code to be used according to the present invention more efficiently to initialize and update display 10. An application program 34 determines the surfaces and the lines to be displayed on monitor 10.

The display monitor 10 (Figure 1), such as a cathode ray tube (CRT) or the like, includes a plurality of pixels which when selectively illuminated define an image being displayed. The number of pixels contained on display 10 will vary depending on the resolution of the display, and in a display as shown with YN x XN pixels, YN and XN are positive numbers which may vary from approximately 400 to 1024. A block 12 includes twelve pixels 14 representative of a block which will have a status word associated therewith.

Figure 2 illustrates three separate blocks 1, 2 and 3 of pixels 14 in the display 10, similar to block 12 (Figure 1). Each of the blocks 1, 2 and 3) includes thirty two pixels 14 and the blocks are adjacent to one another. For convenience, only the first four pixels and the last pixel of each block are shown.

For each block there is a frame buffer 22 and a Z buffer 24. In addition, the present invention utilizes a Za buffer 26 which is a memory capable of storing status information corresponding to each block of pixels. That is, a memory capable of storing a Z value for each block of pixels must be provided either by additional memory, or utilization of existing unused memory. It can be seen that the Za buffer must be able to store at least one Z value for each block of pixels, where the number of blocks will be equal to the number of pixels on the display divided by the number of pixels per block. An understanding of the present invention, in particular the function of the status word Za in maintaining the maximum Z value for the block, will be facilitated by the following description in which these blocks of pixels will be described as a group of linear pixels along a scan line. However, any block of pixels can be utilized, e.g. if the screen has X rows and Y columns then the status word (Za) has X/number of pixels per row and Y/number of pixels per column. This description will use thirty two pixels per block, arranged linearly and parallel to the Y axis of a cartesian coordinate system. It should be noted that in practice the number of pixels per block will be selected to maximize performance based on criteria such as cost and hardware utilization. Any number of rows and columns can be used and the pixels do not have to be arranged linearly.

A line 16 begins within block 1 and ends in block 3, extending from Y1 to Y2. Thus it can be seen that line 16 includes pixels within each of blocks 1, 2 and 3. It should be noted that the present invention contemplates other possible configurations of lines and surfaces to be drawn, as well as other configurations of blocks of pixels.

In the following description of the operation of the present invention with reference to Figures 3A and 3B, it is assumed that the line 16 is to be drawn through the pixels 14 of blocks 1, 2 and 3, as shown in Figure 2.

At step 1, the parameters of the block being considered must be initialized. In this example, the Y values must be extended along the scan line such that all pixels to be drawn as line 16 are considered consecutively, independent of the number of blocks 1, 2 and 3, or the like which may contain the line. For example, in blocks 1, 2 and 3 as shown in Figure 2, the Y value is initially set to correspond to the first pixel in the block containing the beginning of the line being drawn, Y1. In this case, Y1 corresponds to the third pixel in block 1 and Y2 corresponds to the third pixel in block 3. The line to be drawn between Y1 and Y2 includes all of the pixels contained in block 2. In order to process this line three blocks of pixels need to be considered. First, those pixels contained within the first adjacent block 1 including those pixels which occur prior to the beginning of the line being drawn, i.e. pixels 1 and 2 of block 1, and those pixels lying within the line to be drawn, i.e. pixels 3 to 32 of block 1. Second, pixels 1 to 32 of block 2 must be considered when drawing line 16. Third, the pixels of block 3 must also be considered including those pixels associated with line 16, i.e. pixels 1 to 3, and those pixels not associated with the line, but con-

tained within one of the blocks which has at least one pixel within line 16, i.e. pixels 4 to 32 of block 3.

Additionally at step 1, YSTART is set at the left, or beginning pixel of the block being considered, i.e. YSTART is equal to pixel 1. Further, YSTOP is defined as YSTART plus the number of pixels in the block minus one, and equals pixel 32. Any number of pixels can be used for the blocks, but thirty two was chosen to exemplify the present invention and should not be considered a limitation.

Step 2 considers the pixels belonging to an intersected block being drawn, i.e. all pixels of a block having any pixels which are intersected by line 16. If Y is not less than or equal to (i.e. more than) YSTOP, then the pixels cannot be included in the block under consideration and the method proceeds to step 10 (Fig.3B) and ends. It should be noted that YSTART is defined as the first pixel in each block of pixels being considered and YSTOP is defined as the last pixel in each block.

At step 3, it is determined whether the block under consideration is initialized or uninitialized. That is, a determination is made as to whether the pixels contained within the block have previously been written to with a Z value or are "logically cleared", this meaning that Za for the block has a negative value. Specifically, has the Z value for the status word Za for each block been set to a negative value thereby indicating the uninitialized nature of the block of pixels? It should be noted that all blocks having at least one pixel contained in the line being drawn are consi[illegible]d independently of the remaining blocks (subsequently considered) contained on the display 10 regardless of whether a line or surface to be drawn extends into other blocks.

If the blocks 1, 2 and 3 are uninitialized, the processing of block 1 continues to step 4 which determines whether the Y value is less than Y1, the starting point of line 16. It can be seen that pixels 1 and 3 of block 1 satisfy step 4 and the method proceeds to step 5 where the frame buffer for the pixel being considered is written to a background colour (as the line has not started) and the Z buffer and status word (Za) for the block are set equal to the Z max value. Thus, the Z buffer for the pixel now contains a value such that it is initialized. The status word Za is also set to Z max so that a "running total" of the maximum Z value encountered for the pixels in this block is kept. It should also be noted that steps 4 and 5 are looped together by incrementing Y at step 5, i.e. setting Y equal to Y plus one. In this manner, each pixel is considered successively and steps 4 and 5 are continuously repeated while the requirements of step 4 are met.

When Y corresponds to pixel 4 of block 1 it can be seen that Y is no longer less than Y1 and the method proceeds to step 6, where it is determined whether Y is less than or equal to the lesser of YSTOP and Y2. YSTOP is the last pixel in the block 1 under consideration, and Y2 is the last pixel in the line being drawn. In this case, YSTOP for block 1 is pixel 32 of block 1. Therefore, it can be seen that step 6 will be satisfied for pixels 3 to 32 of block 1, because Y will be less than or equal to YSTOP for these pixels. For each pixel where the requirements of step 6 are satisfied, the operation proceeds to step 7 where the frame buffer for the first pixel considered as not less than Y1 is set with a colour value for the line being drawn, the Z buffer is set with the Z value for the line being drawn and status word Za is set to the maximum of the current Za value or the current Z value such that the running total of the maximum Z value for the block is maintained. For each subsequent pixel considered as not less than Y1, the frame buffer is set to the colour value of the previous pixel, plus a delta colour value, determined through interpolation, or the like. That is, the frame buffer for each subsequent pixel is set to a colour value equal to the previous colour value plus the delta colour value. Similarly, the Z buffer for subsequently considered pixels is set to Z (for the previous pixel) plus a delta Z value, also determined through interpolation. It can be seen that for the first pixel in each block delta colour and delta Z will be equal to zero. Therefore, for the first pixel the Z buffer is set equal to the Z value and the frame buffer for the first pixel is set equal to the colour value. Once again, Y is incremented by setting it equal to Y+1 and steps 6 and 7 are repeated for each pixel which satisfies the requirements of step 6 (in this case pixels 3 to 32 of block 1), because the operation loops between steps 6 and 7 during the time when Y is less than or equal to the lesser of YSTOP and Y2.

When Y is no longer less than or equal to the minimum of YSTOP, or Y2, the operation proceeds to step 8 (Figure 3b) which determines whether Y is less than or equal to YSTOP, which was defined with regard to step 1. Thus when the last pixel of block 1 (pixel 32) has been processed, at step 7, and Y has been incremented such that Y=Y+1, Y is not less than YSTOP in step 8. Therefore, the operation continues to step 28 which determines if blocks are remaining to be processed. If so, the operation returns to step 1 (Fig. 3A). At this time, block 1 of Figure 2 has been processed and pixels 1 and 2 have been written with the background colour while pixels 3 to 32 have been written with the line colour. Additionally, the status word Za for block 1 has been set to the maximum Z value which corresponds to any of the pixels 1 to 32 of block 1. That is, Za is set equal to the maximum Z value for any one of the pixels in the block.

Step 1 initializes YSTART for the next block to be considered (block 2) and step 2 determines if Y is less than or equal to YSTOP. Step 3 then determines whether the next block to be processed (block 2) is initialized or uninitialized. Assuming that block 2 is uninitialized, the operation proceeds to step 4 which

then determines if Y is less than Y1. In this case, for block 2, Y is the first pixel in block 2 and is not less than Y1. The operation then proceeds to step 6 which determines if Y is less than or equal to the lesser of YSTOP and Y2. Again referring to Figure 2, it can be seen that Y is in fact less than or equal to the lesser of YSTOP (pixel 32 of block 2) and Y2. The operation then continues to step 7 where the status word Za for block 2 is set to the maximum value of Z, to begin keeping the running total of maximum Z values for the block. For the first pixel in block 2, Za will be set to the Z value of pixel 1 and during consideration of subsequent pixels the status word Za will be set with the maximum of its current value, or the Z value for each pixel satisfying step 6. Additionally, the frame buffer is set to the line colour determined by the delta colour (colour = colour + delta colour), Z is set equal to the Z value determined by delta Z (Z = Z + delta Z) and the Z buffer is set equal to Z (Z buffer is effectively set to Z + delta Z). The pixel being considered is then incremented by Y = Y+1 and iterations between steps 6 and 7 will continue so long as the pixels being considered within block 2 satisfy step 6. It can be seen that all the pixels within block 2 are less than or equal to the lesser of YSTOP and Y2 and each pixel within block 2 will undergo the operations set forth in step 7. After pixel 32 of block 2 is processed at step 7 the operation returns to step 6, where it is then determined that Y is no longer less than or equal to the lesser of YSTOP and Y2, because all the Y values corresponding to the pixels in block 2 have been processed. The operation then proceeds to step 8 where it is determined that Y is not less than YSTOP and the operation continues to step 28 and once again return to step 1. Thus, after the processing of block 2, it can be seen that the status word Za for block 2 is set to the maximum Z value encountered for any one of the pixels 1 to 32 contained in block 2, and that each of the pixels 1 to 32 has been written with the line colour for line 16.

Once again steps 1 and 2 initialize and determine if Y is less than YSTOP, respectively. Step 3 then determines whether the next block to be processed, in this case block 3, is initialized or uninitialized. Assuming that block 3 is uninitialized the operation continues to step 4 where it is determined that Y is not less than Y1. The operation then proceeds to step 6 where it is determined if Y is less than or equal to the lesser of YSTOP and Y2. For block 3, Y2 is less than YSTOP. Pixels 1, 2 and 3 of block 3 (Fig.12) are less than or equal to Y2 and the frame buffer for each of these pixels is written in step 7 with the line colour (colour = delta colour), the status word Za for block 3 is set to the maximum Z value corresponding to any one of pixels 1 to 3 of block 3, the Z buffer is set to the Z value of each pixel, as determined by delta Z, and the Z value is set equal to the Z + delta Z (the delta Z value being the change between the Z value for the previous pixel being considered and the current pixel being considered) such that for each subsequent pixel, the Z buffer will be set to Z + delta Z. Again, steps 6 and 7 are looped and incremented by Y = Y+1 for each pixel satisfying the requirements of step 6. When pixel 4 of block 3 is encountered, Y is no longer less than or equal to Y2, as Y2 corresponds to pixel 3 of block 3, the last pixel corresponding to the line being drawn. Therefore, the operation continues to step 8 where it is determined if Y is less than or equal to YSTOP. Pixel 4 of block 3 is less than YSTOP (4 is less than 32). Therefore, the operation continues to step 9 where the frame buffer for pixel 4 is written to the background colour, the Z buffer is set to the Z max value and Za is also set to the maximum Z value for any pixel in the block, thereby keeping a running total of the maximum Z value for the entire block. Once again, steps 8 and 9 are looped and each pixel satisfying the requirements of step 8 is incremented. In this case pixels 4 to 32 of block 3 are consecutively processed by steps 8 and 9 until step 8 is no longer satisfied. Once pixel 32 of block 3 is processed and incremented by Y = Y+1, then step 8 is no longer satisfied, as the next pixel is included in a different block, and the process continues to step 28 where it is determined if there are any blocks to process. If so, the process returns to step 1 and the next block is initialized. If there are no block remaining to process then the operation proceeds to step 10 and ends.

In the preceding description, the blocks are assumed to be uninitialized. Assuming that blocks 1, 2 and 3 have been initialized as they have been processed as described above, the operation must set the parameters of the block (Y to the left of the block at YSTART) again at step 1. It is then determined whether Y is less than or equal to YSTOP at step 2 and if so, the operation continues to step 3. If however, Y is not less than or equal to YSTOP the operation continues to step 10 and ends. Next, at step 3 it is determined whether the blocks are initialized. Assuming that blocks 1, 2 and 3 have been initialized and considering block 1, the operation will proceed to step 11 where it is determined if delta Z is greater than O, i.e. if the Z slope of the line is away from the front of the screen (line 1, Figure 4). A positive slope (delta Z greater than zero) will indicate that the line to be drawn is getting deeper into the screen when considered from left to right (line 1, Figure 4). A negative slope (delta Z is less than zero) means that the line being drawn is becoming shallower as considered from left to right (line 2, Figure 4). An estimate of the minimum Z value in the block for the line to be drawn is then calculated. If delta Z is positive then the operation proceeds to step 13 and the minimum estimated value of Z for the line ($ZMIN_1$) is merely the initial Z value, as the Z value of successive pixels will increase, i.e. be farther from the front of the screen. If delta Z is negative, then the operation proceeds to

step 12 and an estimate is made of the minimum Z value for any pixel in the block. As in this case, a block contains 32 pixels, the minimum possible Z value for any pixel in the block ($ZMIN_2$) is the initial Z value ($ZMAX_2$) of the first pixel in the block (that is contained within the line) plus the delta Z value, multiplied by 32 (number of pixels in a block). Thus, it can be seen that for lines to be drawn having a positive slope (line 1, Figure 4) the leftmost pixel will have the minimum Z value ($ZMIN_1$) of the line. But, for lines with a negative slop (line 2, Figure 4), the rightmost pixel will have the minimum Z value ($ZMIN_2$) which must be calculated.

Subsequent to step 12 or 13, the operation continues to step 14 where it is determined if the estimated Z value (ZEST) is greater than the Z value within the status word Za for the block being considered. It should be recalled that Za is a representation of the maximum Z value for any pixel within the block and ZEST is a representation of the minimum possible Z value of any pixel in the line being drawn. Comparing these values determines if there is a possibility that part of the line is visible within the block, or if all of the line is "hidden" by the block. Step 14 addresses the hidden line/hidden surface aspect of the present operation. It can be seen that if the estimated minimum Z value for the block of pixels including the line to be drawn, which was determined at step 12 or 13, is greater than the status word Za (maximum Z value) for the block of pixels which have been previously initialized (written to), then all the pixels in the line currently being drawn have a Z value greater than the value of the status word Za of the previously initialized block of pixels. Therefore, the pixels representing the line to be drawn, and contained within the block currently being considered, are obscured from view by the pixels already contained in the previously initialized block (Figure 4). If ZMIN1 and $ZMIN_2$ (Figure 4) are greater than Za, a full block bypass would be implemented during processing of lines 1 and 2. In this case, a full block bypass is implemented at step 15. That is, all the pixels in the block associated with the status word Za are considered to be closer to a viewer than the pixels in the block having the line currently being drawn and corresponding to the previously determined estimated Z value (ZEST). Therefore, the previously initialized block of pixels are considered to "win" and be visible to a viewer of display 10. In a full block bypass it is necessary to advance the colour and Z value of the line to their initial values at the beginning of the next block. It can be seen that implementation of a full block bypass will increase the processing speed, as well as reduce overhead associated with a pixel by pixel comparison of Z values. That is, a single comparison at step 14 (ZEST greater than Za?) may allow a total of 32 pixels (in this example) to be processed. If a full block bypass is implemented the operation then proceeds to step 27 where it is determined if there are blocks remaining to process and if so the method returns to step 1. If there are no blocks remaining to process at step 27 then the method ends at step 10.

If at step 14, ZEST is not greater than the status word Za for the previously initialized block, then a comparison of the Z values at each pixel location must be implemented and is addressed at steps 14A and 16 to 24 (Figure 3b). Za^2 (Figure 4) is not less than ZMIN2, so that a full block bypass cannot be implemented with respect to line 2. In this case a pixel by pixel comparison must be undertaken. In step 14A, the status word Za is set equal to zero because a new maximum value will have to be determined for this block, which will occur during subsequent processing of the pixels at steps 16 to 24.

It can be seen that step 16 is analogous to step 4 and considers those pixels of the block being processed where Y is less than Y1. For example, pixels 1 and 2 of block 1 satisfy step 16, as Y is less than Y1. While Y is less than Y1, the method proceeds to step 17 where the status word Za is set equal to the maximum of the current value of Za, or the Z value within the Z buffer corresponding to that pixel. It should be noted that for the initial pixel being processed at step 17, Za will equal zero and be set to the Z buffer value. For subsequent pixels a comparison between Za and the Z buffer will occur. Again, steps 16 and 17 are looped so that each of the pixels satisfying the requirements of step 16 is incremented by Y = Y+1 and consecutively processed. Once pixel 3 of block 1 is encountered then step 16 is no longer satisfied and the process continues to step 18 which is analogous to the previously described step 6, i.e. is Y less than or equal to the lesser of YSTEP and Y2. Step 18 will be satisfied for pixels 3 to 32 of block 1 and the operation then continues to step 19 where a comparison for each pixel is made. That is, a determination is made of whether the Z value currently stored in the Z buffer, for each pixel, is greater than the new Z value for the line being drawn. If step 19 is satisfied, i.e. Z is less than the value in the Z buffer, the pixel is visible and frame buffer and Z buffer must be written with the new Z value at step 20.

At step 21, it is determined whether Za needs to be updated because of the new value in the Z buffer. If the value in the Z buffer is greater than Ax, then Za is set equal to the new maximum value in the Z buffer at step 22. At step 22A the colour and Z value of the next pixel is computed, based on the interpolated value given by the delta colour and delta Z as previously discussed.

Subsequent to step 22A the operation returns to step 18 which determines whether the next pixel in the block is less than or equal to YSTOP, in a manner as previously discussed.

It can be seen that step 18 will address pixels 3 to 32 of block 1 until all the pixels in block 1 have been processed. At that time, the operation will proceed to

step 23 and as all the pixels in block 1 have been processed, i.e. pixel 32 is not less than YSTOP, then step 27 determines whether there are blocks remaining to process and if so returns to step 1 or ends at step 10. As there are blocks remaining to process (blocks 2 and 3 of Figure 2), the method returns to step 1 and initializes block 2, step 2 determines whether Y is less than or equal to YSTOP and step 3 determines that block 2 has in fact been initialized. Steps 11 to 13 then determine an estimated Z value ZEST for the pixels in block 2, as previously discussed. Step 14 then compared the ZEST with the status word Za for block 2, which has been previously set to a maximum value in step 7. If a full block bypass is called for, then the operation continues to step 27 to determine if there are blocks remaining to process. If a full block bypass is not possible, then after step 14A, step 16 determines if Y is less than Y1 which will not be the case for block 2, as all the pixels contained therein are associated with line 16. Therefore, the operation continues to step 18 which addresses those pixels with Y values that are less than or equal to the lesser of YSTOP and Y2, which will include all of the pixels of block 2. Steps 18, 19, 20, 21, 22 and 22A are looped (incremented by Y = Y+1) and each pixel in block 2 will be consecutively addressed by steps 18 to 22A in the manner as previously described. Once, pixel 32 of block 2 is processed the operation proceeds back to step 18 and on to step 23 as Y has been incremented by Y+Y=1, therefore Y is not less than YSTOP. Again, step 27 then determines if any blocks are remaining to process and returns the process to step 2 for initialization of block 3.

Step 1 initializes block 3, step 2 determines if Y is less than or equal to YSTOP and step 3 determines that block 3 has previously been initialized. Again, steps 11 to 13 calculate an estimated Z value (ZEST) and step 14 determines if a full block bypass is called for. Assuming that a full block bypass is not possible for block 3, the operation continues to step 16 where it is determined that Y is not less than Y1 and proceeds to step 18. It can be seen that for pixels 1 to 3 of block 3, step 18 will be satisfied and these pixels will be processed by the looped steps 18 to 22A, as previously discussed. Once pixel 4 is encountered, step 18 is no longer satisfied and the operation proceeds to step 23, which is similar to step 8, and determines if Y is less than or equal to YSTOP. It can be seen that for pixels 4 to 32 of block 3, step 23 will be satisfied and the status word for block 3 will be set equal to the maximum of the value contained in the Z buffer, or the value currently in the status word Za, whichever is greater. Thus, it can be seen that the Za buffer will contain the maximum value (running total) for any Z value encountered with regard to any pixel contained in the block. Once again, steps 23 and 24 are looped and each pixel satisfying step 23 contained within block 3 will be consecutively processed at step 24. Once the last pixel in block 3 (pixel 32) has been processed and incremented by Y = Y+1, then Y will no longer be less than or equal to YSTOP and the operation proceeds to step 27 which determines if there are any blocks remaining to process. It can be seen that for the described example all the blocks of Figure 2 have been processed and the operation will continue to step 10 and end.

Figures 5A and 5B list pseudo-code which exemplifies one means of implementing the present invention. Once skilled in the art will appreciate the simplicity of the present invention when the pseudo-code is viewed in conjunction with the flowchart of Figures 3A and 3B.

The present invention contains several advantages over conventional Z buffer display systems. For example, less time and overhead is required initially to set the Z values as only the status word Za for a block of pixels needs to be cleared. Conversely, conventional systems must clear each and every pixel individually. Therefore, rendering an object on the display can begin with less delay, thereby improving the efficiency of Z buffer initialization. Additionally, in complex scenes that are often rendered by graphics systems many objects may be hidden by other previously scan-converted objects, such as lines and surfaces. The present invention provides a means of comparing, through a status word Za, a plurality of pixels which have previously been drawn to the display with a plurality of pixels that are to be drawn to the display. This comparison matches the maximum Z value for the currently drawn pixels with the minimum value for the group of pixel to be displayed and determines whether the currently displayed pixels will "win". If so, the currently displayed pixels will remain on the display and another group of pixels to be drawn can then be processed. It can be seen that comparing two groups of pixels with one another saves a great deal of time and overhead when compared with conventional processes which compare each individual pixel contained on a display.

Further, in some cases large portions of the screen may never be used for rendering, i.e. the object may be small and centrally located in the screen. In this case refresh logic can detect the condition where the status word Za was set to a negative number (logically cleared) and insert the background colour for those pixels. Therefore, those areas of the Z buffer which are not used and correspond to portions of the screen for which no scene is displayed, are never actually reset or updated, thus saving additional time when rendering objects on a screen.

Although certain preferred embodiments have been shown and described, it should be understood that many changes and modifications may be made therein without departing from the scope of the appended claims.

Claims

1. A method of displaying at least one object on a computer graphics system having a display with a plurality of pixels thereon and a frame buffer, the method comprising the steps of associating a portion of the plurality of pixels together in a block, storing a maximum depth value for each block, determining whether the object to be displayed intersects the block, and drawing all the pixels within the block to the display.

2. A method of displaying at least one object on a computer graphics system having a display with a plurality of pixels thereon and a frame buffer, the method comprising the steps of associating a portion of the plurality of pixels together in a block, storing a maximum depth value for each block, determining whether the object to be displayed intersects the block, determining an estimated minimum Z value for a block of pixels corresponding to the object to the displayed, comparing the stored maximum depth value for the block of pixels corresponding to the object currently being displayed with the estimated minimum Z value, and continuing to display the current block of pixels when the stored maximum depth value is less than the estimated minimum Z value.

3. A method according to claim 2, wherein the method comprises the steps of comparing the Z values for each pixel in the block corresponding to the object currently being displayed with the Z value for each pixel in the block corresponding to the object to be displayed, and drawing the pixel having the minimum Z value to the display.

4. A method according to claim 1, 2 or 3, wherein the step of storing comprises the steps of providing a memory for each block of pixels, and storing in the memory the maximum depth value of any pixel contained within the block.

5. A method according to claim 2, or any claim appendant to claim 2, wherein the step of determining an estimated minimum Z value comprises the steps of determining whether the slope of the Z values within the block corresponding to the object to be displayed is positive or negative for a positive slope, setting the estimated minimum Z value equal to the Z value of the first encountered pixel in the block corresponding to the object to be displayed, and for a negative slope, setting the estimated minimum Z value equal to the Z value of the first encountered pixel in the block corresponding to the object to be displayed, plus the number of pixels in the block multiplied by a delta Z value.

6. A method according to claim 3, or any claim appendant to claim 3, wherein the step of drawing comprises the steps of writing the frame buffer with a background colour for all non-intersected pixels, and writing the frame buffer with a line colour for all intersected pixels having the minimum Z value.

7. A method according to claim 2, or any claim appendant to claim 2, wherein the step of continuing to display comprises the step of bypassing consideration of the block of pixels corresponding to the object to be displayed.

8. A method according to claim 1, or any claim appendant to claim 1, wherein the step of determining comprises the step of determining whether at least one of the pixels contained within the block is intersected by the object.

9. A method according to claim 8, wherein the step of drawing comprises the steps of writing the frame buffer with a background colour for all non-intersected pixels, and writing the frame buffer with a line colour for all intersected pixels.

10. A computer graphics system having a display with a plurality of pixels thereon and a frame buffer, comprising means for associating a portion of the plurality of pixels together in a block, means for storing a maximum depth value for each block, means for determining whether an object to be displayed intersects the block, and means for drawing the pixels within the block to the display.

11. A computer graphics system having a display with a plurality of pixels thereon and a frame buffer, comprising means for associating a portion of the plurality of pixels together in a block, means for storing a maximum depth value for each block, means for determining whether an object to be displayed intersects the block, means for determining an estimated minimum Z value for a block of pixels corresponding to the object to be displayed, means for comparing the stored maximum depth value for the block of pixels corresponding to an object currently being displayed with the estimated minimum Z value, and means for maintaining the display of the current block of pixels when the stored maximum Z value is less than the estimated minimum Z value.

12. A system according to claim 11, including means for comparing the Z values for each pixel in the block corresponding to the object currently being displayed with the Z value for each pixel in the block corresponding to the object to be displayed and means for drawing the pixel having the mini-

mum Z value to the display.

13. A system according to claim 10, 11 or 12, wherein the means for storing comprises memory means for each block of pixels and means for storing, in the memory, the maximum depth value of any pixel contained within the block.

14. A system according to claim 11, or any claim appendant to claim 11, wherein the means for determining an estimated minimum Z value comprises means for determining whether the slope of the Z values within the block corresponding to the object to be displayed is positive or negative, means for setting, for a positive slope, the estimated minimum Z value equal to the Z value of the first encountered pixel in the block corresponding to the object to be displayed, and means for setting, for a negative slope, the estimated minimum Z value equal to the Z value of the first encountered pixel in the block corresponding to the object to be displayed, plus the number of pixels in the block multiplied by a delta Z value.

15. A system according to claim 12, or any claim appendant to claim 12, wherein the means for drawing comprises means for writing the frame buffer with a background colour for all non-intersected pixels, and means for writing the frame buffer with a line colour for all intersected pixels having the minimum Z value.

16. A system according to claim 11, or any claim appendant to claim 11, wherein the means for maintaining the display of the current block of pixels comprises means for bypassing consideration of the block of pixels corresponding to the object to be displayed.

17. A system according to claim 10, or any claim appendant to claim 10, wherein the means for determining comprises means for determining whether at least one of the pixels contained within the block is intersected by the object.

18. A system according to claim 17, wherein the means for drawing, comprises means for writing the frame buffer with a background colour for all non-intersected pixels, and means for writing the frame buffer with a line colour for all intersected pixels.

19. A system according to any of claims 10 to 18, including application program means for defining the object to be displayed, and central processing means for implementing the actual rendering of the object on to the display.

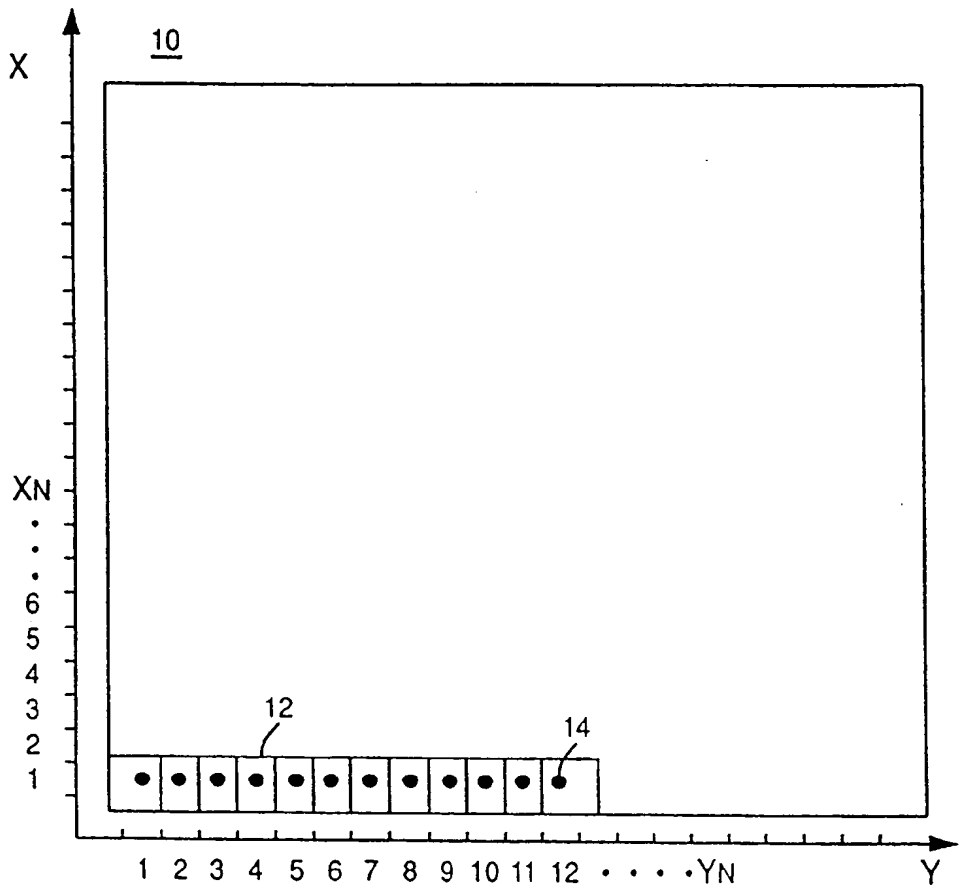

FIG. 1

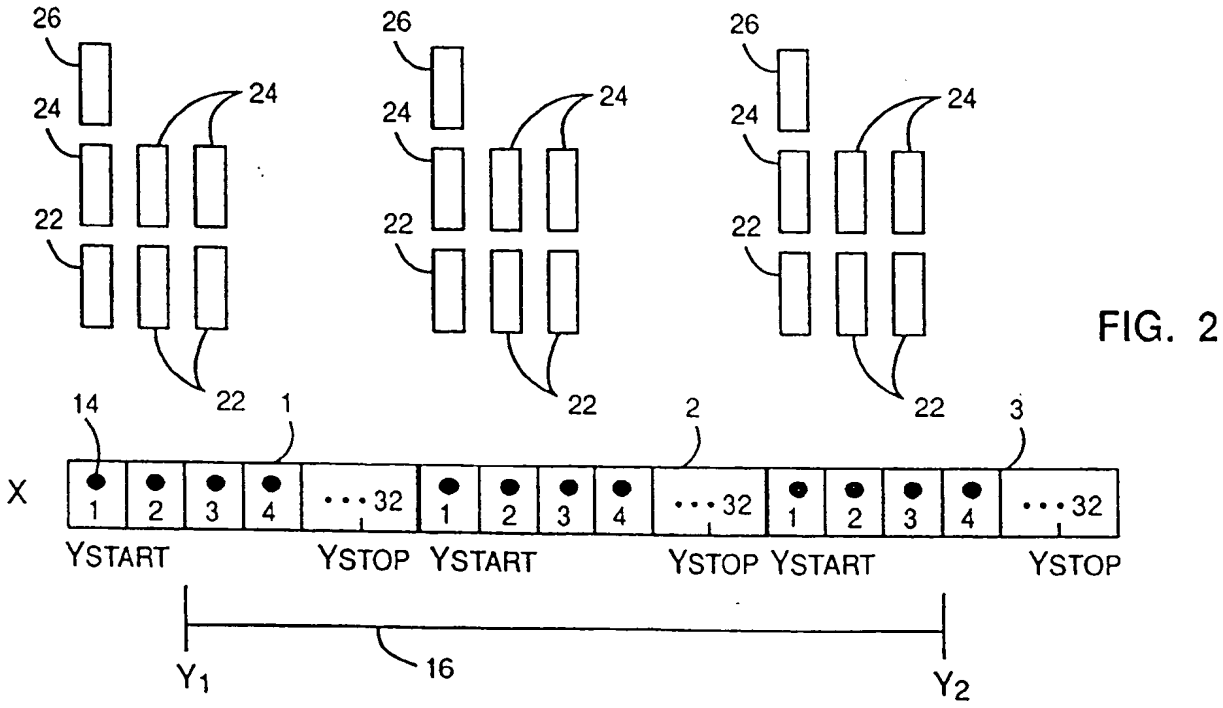

FIG. 2

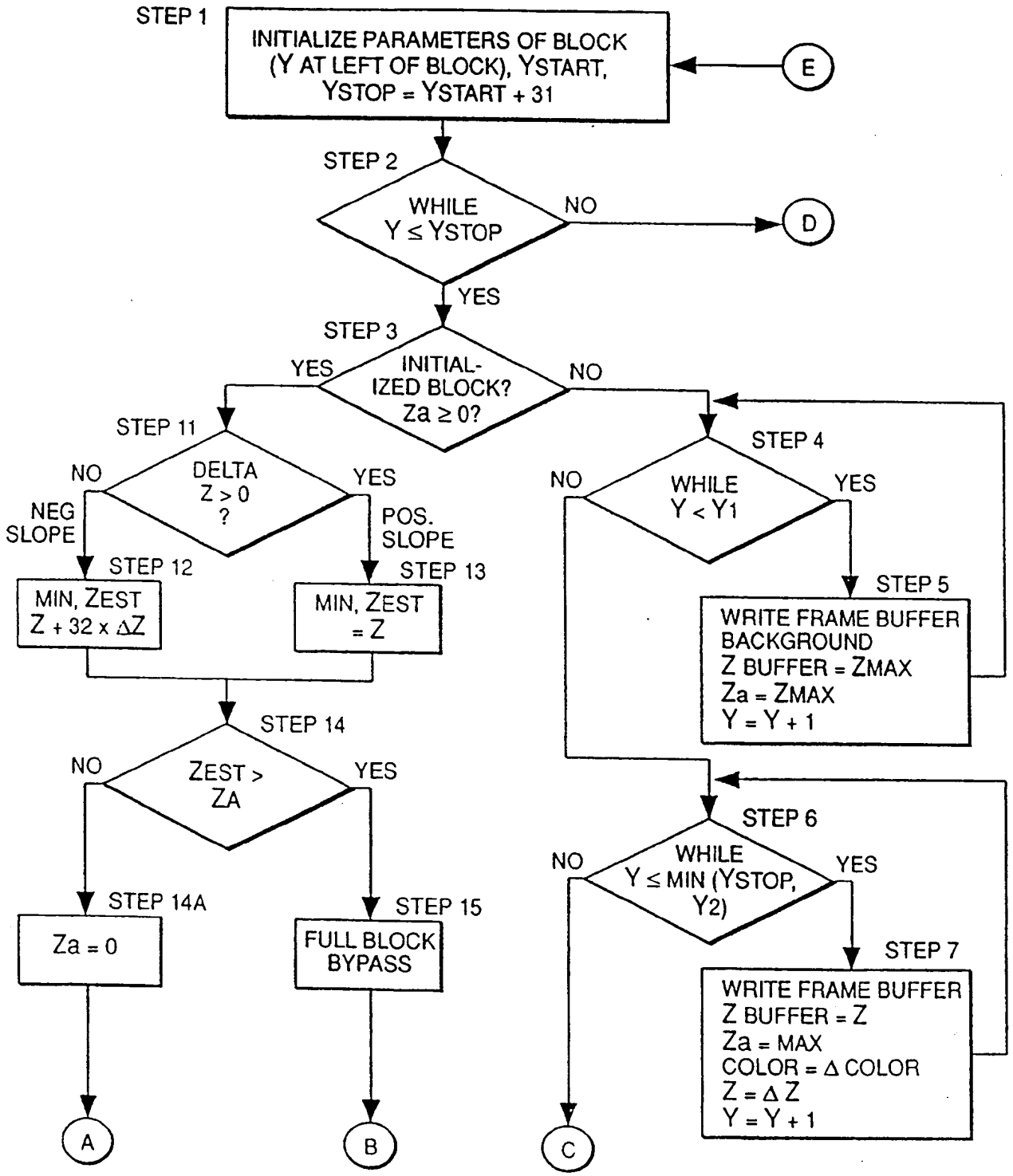

FIG. 3A

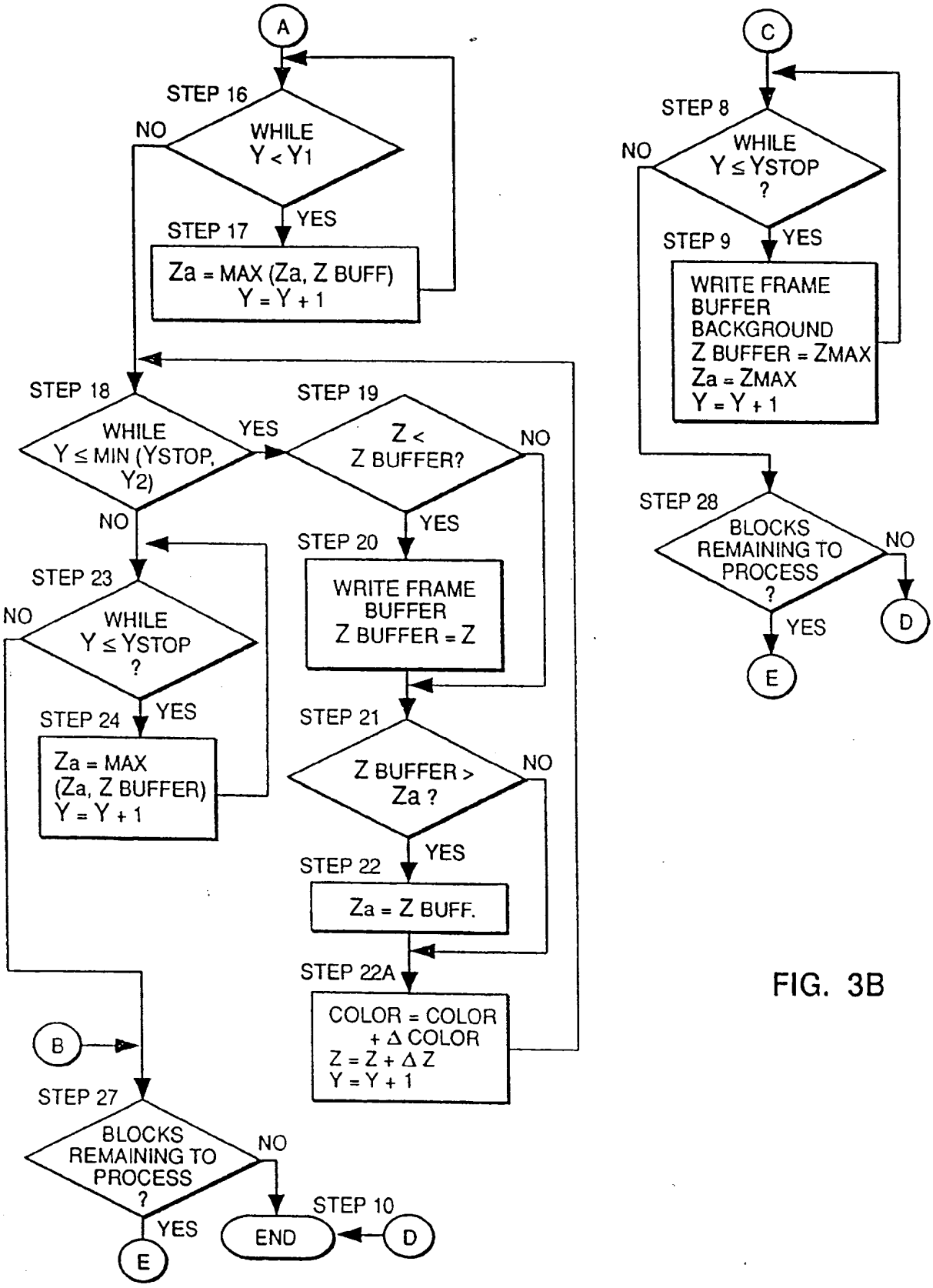

FIG. 3B

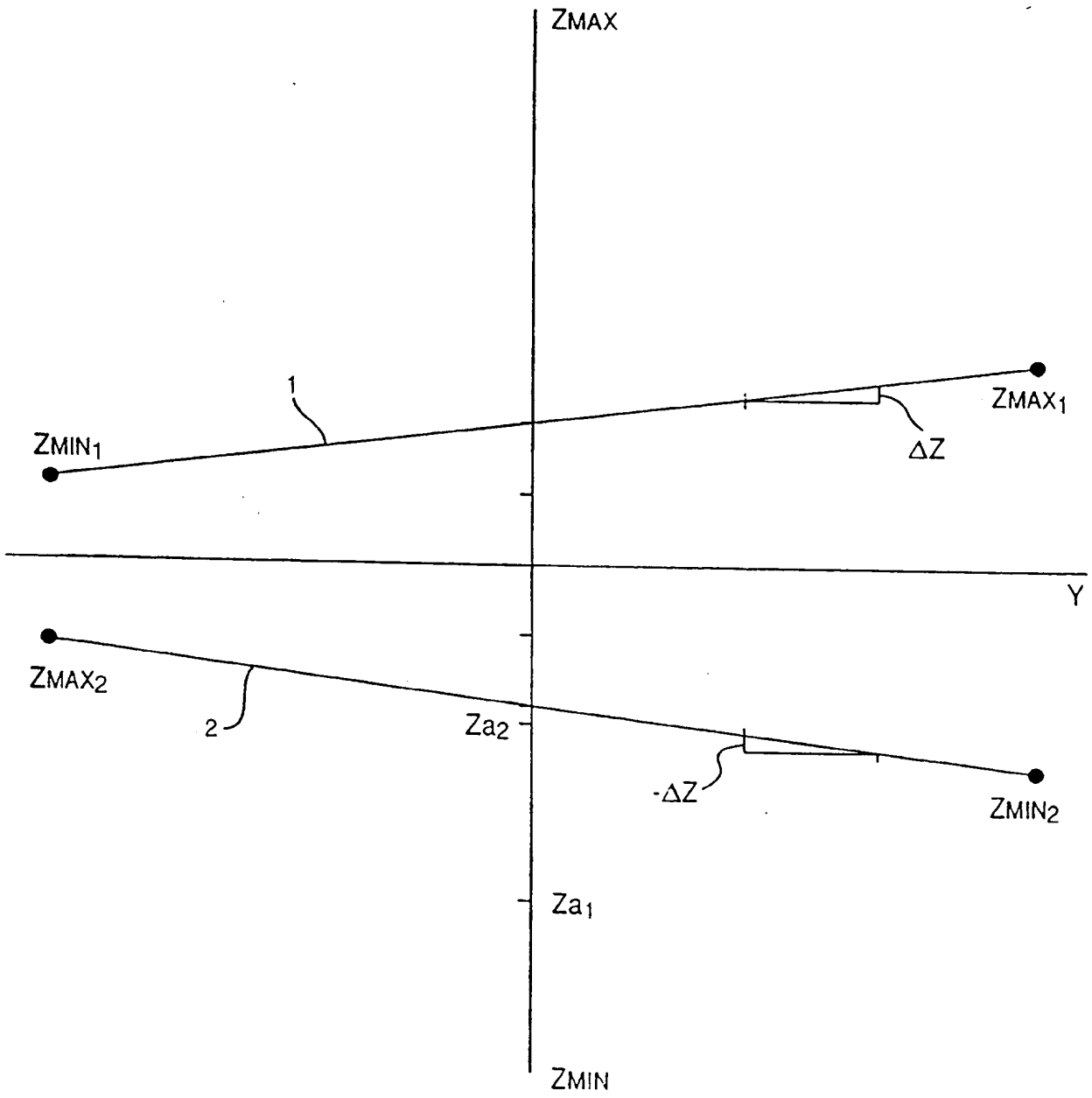

FIG. 4

```
1
2   // Input: scan line from (x, y1) => (x, y2)
3   // Parameters: x, y1, y2, z(x, y1), c(x, y1), delz, delc
4
5   // Expand y extent to a 32 bit boundary
6   y = integer(y1 / 32) * 32;
7
8   while(y <= y2) {
9       ystart = y;
10      yb = y / 32; // za index
11
12      // Uninitialized block Algorithm
13      if(za(x, yb) < 0) {
14          za(x, ya) = 0;   // initialize block maximum
15
16          // BEFORE
17          while(y < y1) {
18              write_frame_buffer(x, y. BACKGRD);
19              zbuf(x, y) = ZMAX;
20              za(x, yb) = ZMAX;
21              y++;
22          }
23
24          // DURING
25          ystop = min(ystart + 32, y2);
26          while(y < ystop) {
27              write_frame_buffer(x, y, c);
28              zbuf(x, y) = z;
29              za(x, yb) = max(za(x, yb), z);
30              c += delc;
31              z += delz;
32              y++;
33          }
34
35          // AFTER
36          while(y < ystart + 32) {
37              write_frame_buffer(x, y, BACKGRD);
38              zbuf(x, y) = ZMAX;
```

FIG. 5A

```
39              za(x, yb) = ZMAX;
40              y++;
41          }
42      } // end uninitialized block algorithm
43
44      // Initialized block algorithm
45      else {
46          // scan line minimum in this block
47          if(delz > 0)  zest = z;
48          else zest = z + 32 * delz;
49
50          if(zest > za(x, yb)) {   // bypass block
51              if(y1 < y) {  // full block bypass
52                  y += 32;
53                  c += 32 * delc;
54                  z += 32 * delz;
55              }
56              else {  // partial block bypass
57                  while(y < ystart + 32) {
58                      y++
59                      c += delc;
60                      z += delz;
61                  }
62              }
63          } // end of block bypass
64
65          else {  // need to scan convert
66              za(x, ya) = 0;  // initialize block maximum
67
68              // BEFORE
69              while(y < y1)  {
70                  if( zbuf(x, y) > za(x, yb) ) za(x, yb) = zbuf(x, y);
71                  y++;
72              }
73
74              // DURING
75              ystop = min(ystart + 32, y2);
```

FIG. 5B

```
76              while(y < ystop) {
77                  if(z < zbuf(x, y) ) {
78                      write_frame_buffer(x, y, c);
79                      zbuf(x, y) = z;
80                  }
81                  if( zbuf(x, y) > za(x, yb) ) za(x, yb) = zbuf(x, y);
82                  c += delc;
83                  z += delz;
84                  y++;
85              }
86
87              // AFTER
88              while(y < ystart + 32) {
89                  if( zbuf(x, y) > za(x, yb) ) za(x, yb) = zbuf(x, y);
90                  y++;
91              }
92          } // end of block scan convert
93
94      } // end of initialized block algorithm
95
96  } // end of scan line while
97
```

FIG. 5C

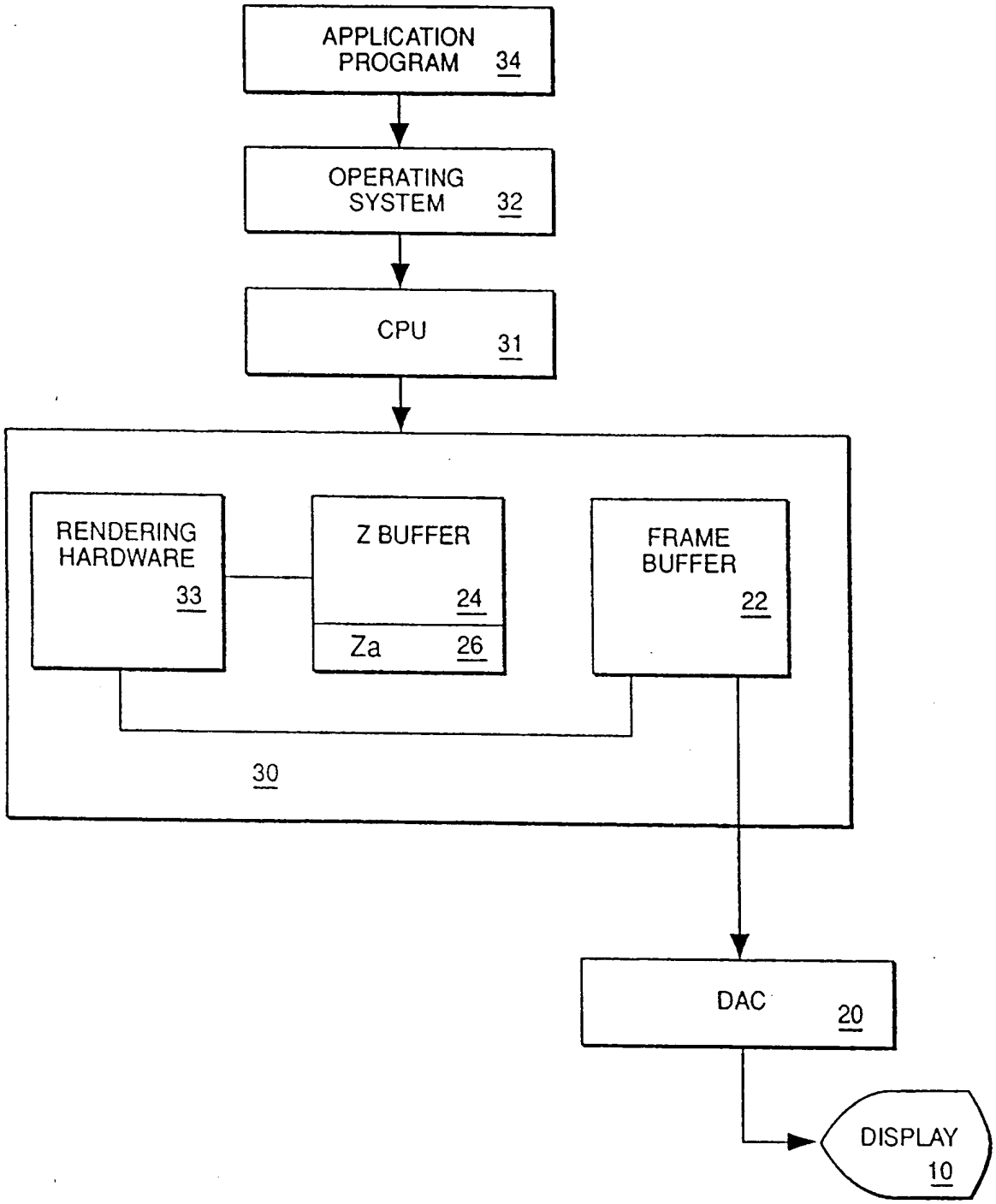

FIG. 6